Project requirements for 34359

Objectives

The aim of the project is that you design and implement an SDN-based network service on your own, based on the tools and mechanisms we have seen during the lectures.

Requirements

As a "product" of this process, you are expected to deliver a report and to do a presentation to the rest of the class, covering the following elements:

1. **Problem Statement:** what is that your service does, why, context and expectations, from a generic networking perspective.
2. **Project Design:**
    a. Project Description: technical description of your SDN-based solution: the mapping of the previous generic statement to concrete elements and technologies providing a solution to it.
    b. Involved entities and network relations (functional and topological distribution).
    c. Overall service behavior description, including:
        1. Cases.
        2. Chronogram of relevant message–exchanges, among entities.
        3. Expectations-Impact description.
3. **Project Implementation:**
    a. Description and details of involved software elements and their functionality according to the previous design.
    b. Code-Flow Diagrams (the source and compiled modules/applications should be provided, although not "in paper").
4. **Project Testing strategy.**
5. **Results Analysis.**
6. **Critical Conclusion.**

While the **presentation** contents are supposed to be illustrative, the report contents must be detailed. This includes references to source code (consider 3.b above). All the source code is expected to be included, in electronic version, as a byproduct for the report, together with compiled binaries and installation instructions.

The duration of presentations should be 15 minutes, including demonstration of the service operation. A 5 minutes gap for questions will follow each presentation.

The corresponding **report** (and corresponding electronic material, including an electronic copy of the presentation and the report) should be handed in, **latest, the 16th of January, at 23:59 (11:59 p.m.)**.

Time and place

Presentations will take place on the **11th of December or 18th of December** (Two Wednesday), depending on the final number of projects. Please tell in the email 1(most liked) to 5 (less liked) and also the time that you want (11th or 18th of Wednesday). Please book your presentation, **before 27th October (23:59)**. To do so, please use the Forum created in DTU Learn, so that others can see the topics that have been chosen-booked

already. Send me also an email, so that I keep a copy of your project booking. Note: a topic can be chosen by different students/groups a maximum of 2 times.

And it can be chosen 2 independent people (also 2 people and 2 people). For instance, maybe the first email and the second email, of course independent, want to do project 5 (Host tracking system). The third if he wants to so number 5, it cannot be done unfortunately. That is why I ask you to send me several options, because if the option 1 cannot be done so option number 2 might be possible.

Also, if you do something for the presentation, you can enhance it for the final report.

For instance you have to send us:

You projects proposal (you include option 1,2,3,4 and 5)

- Option 1: IPv4 NAT service
  (…)´
  Option 5: Implement a L2 mac learning mechanism together with VLAN support.

- 11th of December is my presentation day.
- We are 1 or 2 people (the full name)

In summary:

Presentation: 11th December or 18th of December

Presentation book: before 27th of October

Report: latest 16th of January

Possible services / topics

The following are examples of ideas related to likely services. Some of them can be worked-out as teams of max. 2 people. This is indicated accordingly. If you have any different service/project idea, which you would like to discuss with me, please drop me an email with details, for approval.

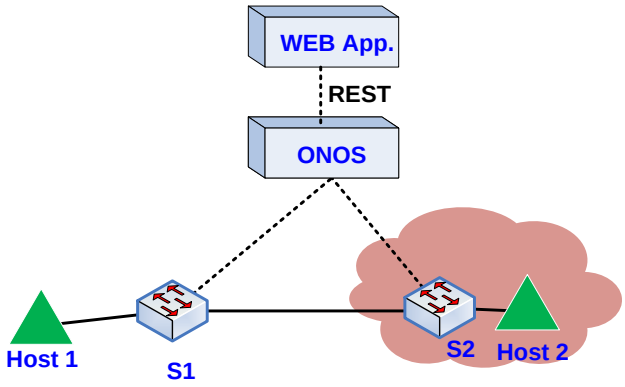
1) Implement a **monitoring application**. The application is supposed to show on a web interface, all the links in the network, the link capacity and the link utilization. Further improvements can be made so that the application shows also the traffic types travelling along each link. (suitable for team work)

2) **Implement a L2 mac learning mechanism together with VLAN support**. The final proof-of-concept should be able to segregate traffic into at least 2 VLANS. We can assume that all the hosts that connect to odd switch-ports belong to VLAN 1 and the hosts connected to even-ports belong to VLAN 2. Assuming a single subnet (e.g. 10.0.0.0/24), any host should be able to ping only the hosts that are on the same VLAN. The forwarding should be based on MAC learning: each switch will learn what MAC addresses have been

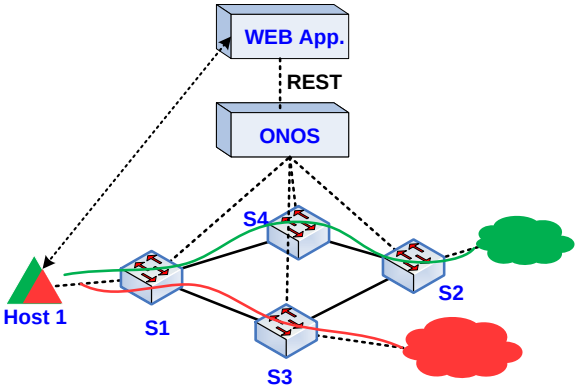
seen on each of its ports and the corresponding VLAN ID of those MAC addresses. Forwarding will be done based on dest MAC + VLAN. For example, an Openflow rule in the switch would look like this:dest_mac = 00:00:00:00:00:01, vlan= 2, output port 3. If a host is trying to reach 00:00:00:00:00:01, it will only be able to reach it if it also matches the VLAN tag (they are on the same VLAN), otherwise the rule won't match and the packet will be dropped (or sent to the controller and then dropped explicitly using the PacketOut OF message). As FURTHER implementation, it should be put on DeviceLearning and HostLearning application. When a new host appears it should post (proactively) the go/back learning OpenFlow rules. If the host is odd to belong to VLAN 1 and even belong to VLAN 2.

a. Go: matching MAC/VLAN (1 or 2), action output host
b. back matching MAC/VLAN (1 or 2) output switch rule (it could be more than one MAC/VLAN)

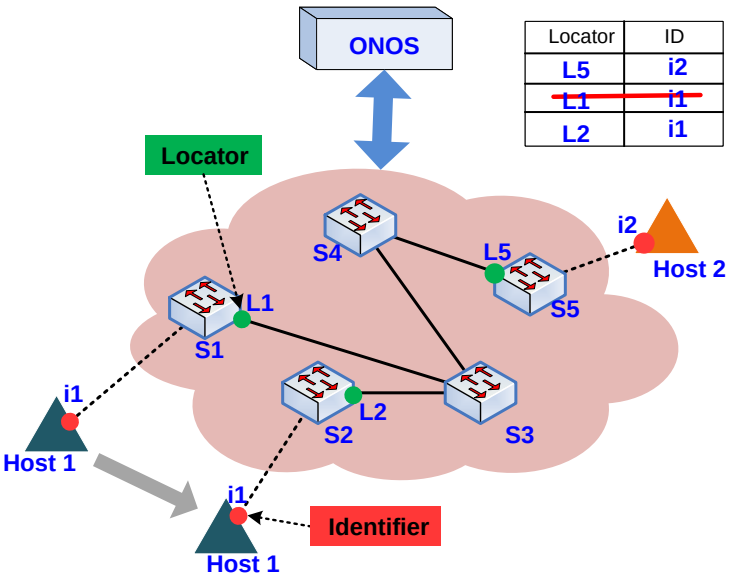
3) Consider the scenario illustrated in the figure below. The idea of this project is to implement an application which **monitors the bandwidth utilization on S1 for Host 1**. The host is allowed to use the whole available bandwidth until it reaches a certain limit. When the limit has been reached then the application will cap the traffic of the host up to a specific level (for instance from 1 mb/s to 500 kb/s). It may completely stop the traffic too (that easier than the first one). The solution can be based on having two queues in S1. The first queue is a normal queue without any QoS parameters enforced on it. The second queue will be a rate limiting queue. User's traffic is normally forwarded using the first queue and monitored. When the limit has been reached, the traffic is forwarded using the second queue which limits the bandwidth for the user. (suitable for team work) Please find "ONOS QoS" in Google.



4) **Multi-VPN home gateway application**. In this scenario there is a host which may want to connect to multiple VPNs. The same scenario may apply also for multi-homing: one customer with multiple connections to different Service Providers (SPs). Implement an application through which the host can manage its VPNs, connect to any of the available VPNs. The application will configure the network using OF so that the traffic from that moment on, sent through the home gateway will be forwarded to the specific VPN. There's no need to do tunneling for this solution although it is also possible to implement it with tunnels. The application can be an ONOS Application or an external application using ONOS REST API, as in the figure. (suitable for team work)



5) **Host tracking system**. Assume that each host has a fixed identifier (ID). No matter where the host attaches in the overall network, it still keeps its own ID. In the core of the network each device has associated a Locator. The locator helps in routing packets from one part of the network to another (it helps to locate the destination, but it does not identify the destination). The controller keeps track of each host in the system by having a mapping between IDs and Locators. In the case that host 2 wants to send traffic to host 1, the initial packet will have as a destination the ID of host 1 (i.e. i1). At the edge of the network (at S5) the controller will change the addressing to use locators, so the packet is forwarded from L5 to L1. At S1 the identifiers are put back again so that the final host can be reached in case multiple hosts are attached to S1. The best way to achieve this is to add the locator headers on top of the identifiers at the ingress and strip the locator headers at the egress. If host 1 moves to a new part of the network it still keeps i1 as an ID but the locator will be different in the centralized mapping at the controller. The routing in the core (based on locators) can be done in any possible way, but it would be nice if it is a dynamic solution, instead of using static routes for a fixed topology. Note: Mininet Wifi (https://github.com/intrig-unicamp/mininet-wifi/wiki could be an interesting option to ease the work with this project. (suitable for team work)



6) **Energy optimization module**: routing according to an objective function. An example of an objective function can be to map all traffic to as fewer routes as possible in order to be able to switch-off devices and/or ports in the network.

7) **Network survivability module**: the service supports computation of disjoint paths in the network. The paths should be installed in the OpenFlow devices such that, in case of a link/node failure, the time to switching to an alternative path is reduced to a minimum. Investigate *Fast Failover* mechanisms.

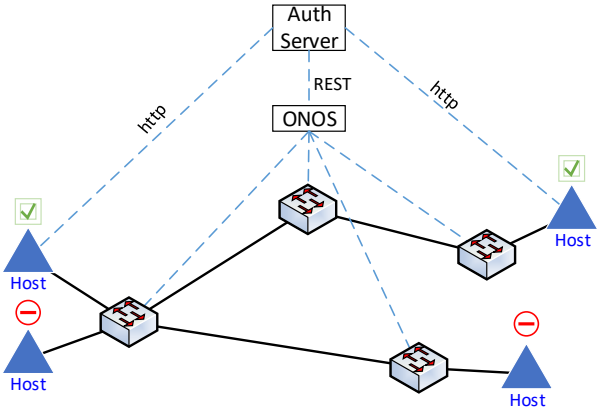
8) **Security mechanism**: (suitable for team work). It will be from proxyARP, custom ReactiveForwarding, and custom proxyARP, all separated projects. I will explain now what they are. Imagine both Host 1 and Host 2 the ping each other. Imagine the Host 3 and Host 4 comes alive (so you will have used the intent_topo.py on Voluntary Exercise 5). The issue is that when a Host 3 and Host 4 comes to the same switch (Switch 1 for Host 1 and Switch 2 for Host 2), and ping each other, it will won't will be discarded because they have the same MAC as Host1 And also Hots 2 and Host 4. For instance, 00:00:00:00:00:01 and No VLAN of Host 1 and Host 3.
    a. First, use proxyArp, the one that already comes with ONOS. See if that works by default. If it is then don't do the third examples. See if they have to use "more time" for the maps (OsgiPropertyConstants.java).
        i. https://github.com/opennetworkinglab/onos/blob/master/core/net/src/main/java/org/onosproject/net/OsgiPropertyConstants.java
    b. The second app (you have to deactivate proxyArp) it will be with ReactiveForwading (fwd). when a packet comes with ARP, it will be accept and put it in a map. And answer with the correct IP, MAC and so on. It will be good if you Wireshark the h1-eth0 if you check which IP and MAC the proxyARP has. The code in Lecture 6 for the "LB service" is a good example (slide 12).
        i. https://github.com/opennetworkinglab/onos/blob/master/apps/fwd/src/main/java/org/onosproject/fwd/ReactiveForwarding.java
    c. The third additional example (when proxyARP does not work, but you don't have to do it). It will be custom proxyARP (you will use the "onos-create-app" from Lecture 3 and on). It will use the same Maps and and discard some packets if they don't come from legitimate users.
    d.
        i. https://github.com/hsjts0u/ProxyArp/blob/main/src/main/java/nctu/winlab/ProxyArp/ProxyArp.java

9) **IPv4 NAT service**: Implement network/port address translation. It will be implemented as a controller application/module. The service will support basic IPv4 bidirectional address and port translation with an OpenFlow based switch. The translations must be able to be configured by the user.

10) Extend the **load balancer service** presented in lecture 6 in the course, so that it works with different topologies and different balancing strategies / scenarios, for example .

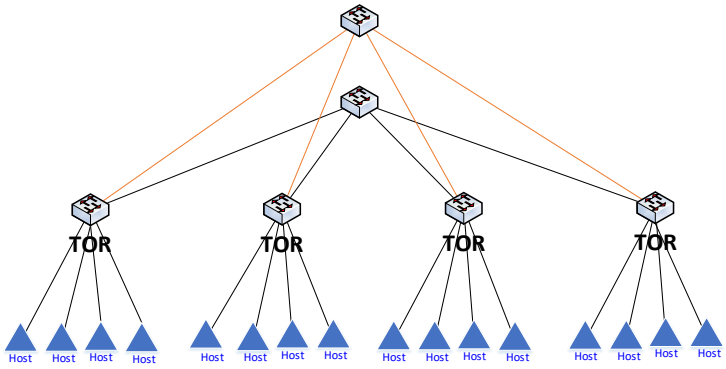
11) **Authentication Service**. (suitable for team work). Before any host can send (or receive) network traffic, each host must first authenticate with an authentication service, via a web interface. The web interface is associated to a web application managing the authentication operation, as well as communicating to ONOS via its REST API. Your project should follow the following workflow:
    1. Initially all hosts are disabled to transmit / receive, by the network.
    2. In order to be authenticated, a host user should visit a web interface and enter his credentials. The web interface should submit the info to a web server running a web application, to verify the credentials:
        a. If the credentials are invalid the web server should return an ERROR message.

   b. If the credentials are valid the web server should issue a REST call to the SDN Controller containing the ID of the host.
3. Upon reception of a REST call from the web application, the SDN controller should then enable connectivity in the network for the corresponding host.
4. The web application should reply to the user, via its web interface of the result of the operation.



12) **"Optical" Bypass.** The following topology represents a data centre in which the different hosts are connected between them in a number of ways.
1. Local hosts are connected directly though their top of the rack switch (TOR).
2. Remote hosts are connected.
   a. Through the electrical plane, through a multi-hop path (black).
   b. Through an electrical – optical plane, through a multi-hop path(red).

Your application should forward traffic in any of these three ways depending on the traffic scenario.
1. Local traffic should never leave the TOR.
2. Remote traffic that meets pre-defined criteria (e.g. source destination IP addresses and/or transport ports) should be sent (if possible) through the optical bypass. If the optical bypass is occupied the traffic should be sent thought the electrical only plane.
3. Remote traffic that does not meet the criteria should be sent though the electrical-only plane.



Important Notes:
1. You can emulate the optical plane with very fast, low latency connections.
2. Flows should not be able to share optical circuits. When a port pair in the "optical" switch is occupied by a flow, no other flow should be able to use any of these two ports.